

Logic Programming Engineering – Assignment 4

Dr.-Ing. Sascha Klüppelholz

Exercise 4.1 [Reimplementation of list operations]

Reimplement the following list operations and add all predicates to `mylist.prolog`.

- a) Provide a Prolog predicate for computing/ deciding the length of a given list.

Example:

```
?- mylist_length([a,b,c,x,d],X).  
X = 5.
```

- b) Write a predicate `mylist_member/2` that decides membership of an element to list.

Examples:

```
?- mylist_member(x,[a,b,c,x,d]).  
true.  
  
?- mylist_member(x,[a,b,c,d]).  
false.
```

- c) Write a predicate `mylist_occurrences/3` that counts the number of occurrences of an element within a list.

Examples:

```
?- mylist_occurrences(x,[a,b,c],N).  
N = 0.  
  
?- mylist_occurrences(x,[a,b,x,c,x],N).  
N = 2.
```

- d) Write a predicate `mylist_last/2` that returns the last element of a list.

Examples:

```
?- mylist_last(X,[]).  
false.  
  
?- mylist_last(X,[a,b,c]).  
X = c.
```

- e) Provide your own predicate `mylist_element_at/3` which returns the *n*-th element of a list. The index of the head of a list should be 1.

Examples:

```
?- mylist_element_at([a,b,c],2,X).  
X = b.
```

```
?- mylist_element_at([a,b,c],0,X).  
false.
```

- f) Write a Prolog predicate `mylist_add/3` that allows adding an element to a list without introducing duplicates. Check what Prolog returns for the following query: `mylist_add(a,[a,b,c,a],[a,a,b,c,a])`.

Examples:

```
?- mylist_add(x,[a,b,c,a],L).  
L = [x, a, b, c, a].  
  
?- mylist_add(a,[a,b,c,a],L).  
L = [a, b, c, a].
```

- g) Provide a Prolog predicate `mylist_remove/3` that allows to remove all occurrences of an element from a given list.

Examples:

```
?- mylist_remove(x,[a,b,x,x,c],L).  
L = [a, b, c].  
  
?- mylist_remove(x,[a,b,c],L).  
L = [a, b, c].
```

- h) Provide a predicate `mylist_remove_duplicates/2` for removing duplicates from list.

Examples:

```
?- mylist_remove_duplicates([a,b,c],L).  
L = [a, b, c].  
  
?- mylist_remove_duplicates([a,b,c,a,b,c,b,b],L).  
L = [a, c, b].
```

- i) Write a predicate `mylist_replace/4` that allows replacing all occurrences of an element in list with another element.

Example:

```
?- mylist_replace([a,b,c,a,b],b,d,L).  
L = [a, d, c, a, d].
```

- j) Provide a predicate `mylist_concat/3` for list concatenation.

Example:

```
?- mylist_concat([a,b],[c,d],L).  
L = [a, b, c, d].
```

- k) Write a Prolog predicate `mylist_reverse/2` for reversing a list. Define a predicate `mylist_palindrome/1` on the basis of `mylist_reverse/2` that decides whether a given list is a palindrome.

Example:

```
?- mylist_reverse([a,b,c,d],L).  
L = [d, c, b, a].
```

Look at the internal implementation within SWI-Prolog of the above predicates by using `listing/1`.

Exercise 4.2 [Powerset]

Provide a predicate `mylist_power/2` for creating the powerset of a list (interpreted as set).

Example:

```
?- mylist_power([a,b],P).  
P = [[], [b], [a], [a, b]].
```

Exercise 4.3 [List permutations]

Provide a predicate `mylist_permutation/2` that enumerates all permutations of a list.

Example:

```
mylist_permutation([a,b,c],L).  
L = [a, b, c] ; L = [a, c, b] ; L = [b, a, c] ; L = [b, c, a] ; L = [c, a, b] ;  
L = [c, b, a] ; false
```